UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/717,941	11/20/2003	Stephen La Roux Blinick	TUC920030135US1	9013

45216 7590 10/01/2008

Kunzler & McKenzie
8 EAST BROADWAY
SUITE 600
SALT LAKE CITY, UT 84111

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
10/01/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/717,941	BLINICK ET AL.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *the amendments dated7/29/2008; 8/28/2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-7,9-26 and 28-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-7, 9-26, and 28-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on August 28, 2008 has been entered.

2. Applicant's amendments dated July 29, 2008 and August 28, 2008 respectively, responding to the May 29, 2008 Final Office action provided in the rejection of claims 1-30, wherein claims 1, 5, 10-11, 13, 18-20, 24, 26, and 29-30 have been amended and claims 8 and 27 were canceled.

Claims 1-7, 9-26, and 28-30 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are moot in view of the new grounds of rejection – see *Rocray et al.* - art made of record, as applied here

Claim Rejections – 35 USC § 101

35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

3. Claims 1-7, 9, 10-12, and 29 are rejected under 35 U.S.C 101 because the claims are directed to non-statutory subject matter.

4. **As to claim 1**, "An apparatus for updating a code image, comprising ... an loader ... a logic module ... a bootstrap module ... a copy module ..." is being cited (underline emphasis added); however, it appears that the loader, the logic module, the bootstrap module, and the copy module (recited on paragraph [011], [055] in the specification) would reasonably be interpreted by one of ordinary skill in the art as computer listings per se, are not physical "things". They are neither computer components nor statutory processes, as they are not "act" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer which permit the computer program's functionality to be realized.

In contrast, a claimed computer readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of the computer which permit the computer program's functionality to be realized, and is thus statutory. Accordingly, it is important to distinguish claims that define descriptive material per se from claims that define statutory inventions. (See MPEP 2106.01(I))

5. **As to claims** 2-7 and 9, they do not cure the deficiency of base claim 1, and also are rejected under 35 U.S.C. 101 as set forth above.

6. **As to claim 10**, please refer to claim **1** above, accordingly.

7. **As to claims** 11-12, they do not cure the deficiency of base claim 10, and also are rejected under 35 U.S.C. 101 as set forth above.

8. **As to claim 29**, "An apparatus for updating a code image, comprising ... means for loading a new code image... means for identifying incompatibilities ... means for reconciling the incompatibilities ... means for copying the new code image ..." is being cited (underline emphasis added); however, it appears that the means for loading a new code image, the means for identifying incompatibilities, the means for reconciling the incompatibilities, and the means for copying the new code image (recited on paragraph [011], [055] in the specification) would reasonably be interpreted by one of ordinary skill in the art as computer listings per se, are not physical "things". They are neither computer components nor statutory processes, as they are not "act" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer which permit the computer program's functionality to be realized.

In contrast, a claimed computer readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of the computer which permit the computer program's functionality to be realized, and is thus statutory. Accordingly, it is important

to distinguish claims that define descriptive material per se from claims that define statutory inventions. (See MPEP 2106.01(I))

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1-4, 7, 10, 13, 20-22, 26, and 29-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Talati (Pub. No. US 2004/0044997 A1) (hereinafter 'Talati') in view of Rocray et al. (WO 02/13003 A2) (hereinafter 'Rocray' - art made of record)

10. **As to claim 1** (Currently Amended), Talati discloses an apparatus for updating a code image (e.g., Fig. 2), comprising:

- a loader configured to load a new code image (e.g., [0014], lines 2-3; Fig. 2, element 102; [0047], lines 1-2) into a temporary memory location (e.g., Fig. 1, element 108; [0009], staging area) separate from a memory space (e.g., Fig. 1, element 110; [0009], line 2, runtime area) occupied by and used by an old code image (e.g., Fig. 2, elements 218, 110; [0046], lines 1-8; [0047], lines 1-2); and

- a copy module configured to copy the new code image into the memory space occupied by the old code image (e.g., Fig. 2, element 202; Fig. 3, element 302; [0013], lines 1-3)

Further, Talati discloses a method and apparatus for downloading a new version of an executable code onto a system without the need to bring the system to a halt, thereby sacrificing system up time (e.g., [0005]) but does not explicitly disclose a logic module configured to identify incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; a bootstrap module, within the new code image, configured to reconcile incompatibilities by changing an initialization order.

However, in an analogous art of *system and method for implementing a self-activating embedded application*, Rocray discloses:

- a logic module configured to identify incompatibilities (e.g., P. 1, 1st Para - ... dynamically resolving compatibility issues between the different components in the system ...; 3rd Para - ... comprises a software generic control information file (SGC) ... relating to the compatibility of the software components ...) between the old code image and the new code image from version information (e.g., P. 7, Lines 11-14 - ... identifies which software version ...; P. 8, 2nd Para - ... their internal indication of minimum configuration version updated to reflect their incompatibility with the table format identified by the new configuration ...; P. 9, Table 3 – SGC: Component Level Information, the entry of 'version'; P. 10, 1st

Para – The version field defines the version of the component), a difference in initialization requirements (e.g., P. 7, Lines 13-14 - ... for determining which component are to be used at startup ...; P. 11, Lines 6-8 – If such a component were modified through a partial product software update, the whole system would be re-initialize ...), and a difference in size and location between the old code image and the new code image (e.g., P. 9, Table 3 – SGC: Component Level Information, the entry of 'size');

- a bootstrap module, within the new code image, configured to reconcile incompatibilities by changing an initialization order (e.g., P. 15, Lines 5-8 - Boot code section - ... the boot code of a given processor module ...; P. 19, Lines 6-9 - ... to instruct the system that the alternate bank should be treated as the active bank at the next boot up ...; Lines 12-17 - ... as the result of a software upgrade ... enables the processor modules affected by the update to initialize themselves and boot up)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rocray into the Talati's system to further provide a logic module configured to identify incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; a bootstrap module, within the new code image, configured to reconcile incompatibilities by changing an initialization order in Talati system.

The motivation is that it would further enhance the Talati's system by taking, advancing and/or incorporating the Rocray's system which offers significant advantages that a system and method for dynamically resolving compatibility issues between the different components in the system as once suggested by Rocray (i.e. P. 2, 1st Para)

11. **As to claim 2** (original) (incorporating the rejection in claim 1), Talati discloses the apparatus wherein the old code image is updated substantially concurrent with normal execution of transactions by the apparatus (e.g., [0006]; [0008], lines 5-12]; [0011])

12. **As to claim 3** (original) (incorporating the rejection in claim 1), Talati discloses the apparatus further comprising an initialization module configured to initiate execution of a run-time segment of the new code image (e.g., [0014])

13. **As to claim 4** (Previously Presented) (incorporating the rejection in claim 1), Talati discloses the apparatus wherein the copy module copies the new code image into the memory space (e.g., Fig. 2, Copier copies new code; Fig. 3, element 302)

And, Rocray discloses after the bootstrap module reconciles incompatibilities (e.g., P. 15, Lines 5-8 - Boot code section - ... the boot code of a given processor module ...; P. 19, Lines 6-9 - ... to instruct the system that the alternate bank should be treated as the active bank at the next boot up ...; Lines 12-17 - ... as the result of a

software upgrade ... enables the processor modules affected by the update to initialize themselves and boot up)

14. **As to claim 7** (Previously Presented) (incorporating the rejection in claim 1), Talati discloses the bootstrap module recognizing persistent data associated with the old code image and associates the persistent data with the new code image such that the persistent data is available in response to execution of the run-time segment of the new code image (e.g., [0012] – the conversion module provides two separate functionalities (a) to selectively reconcile incompatibilities between the old code image and the new code image cited in claim 1; and, (b) to recognize persistent data described in this claim)

15. **As to claim 10** (Currently Amended), Talati discloses an apparatus for updating a code image (e.g., Fig. 2, copier copies new code), comprising of an update module configured to load a new code image (e.g., [0014], lines 2-3; Fig. 2, element 102; [0047], lines 1-2) into a temporary memory location (e.g., Fig. 1, element 108; [0009], line 2, staging area) separate from a memory space occupied by and used by an old code image (e.g., Fig. 2, elements 218, 110; [0046], lines 1-8) and a bootstrap module within the new code image that executes subsequent to the update module (e.g., Fig. 2, element 202; [0047], lines 1-2)

Further, Talati discloses a method and apparatus for downloading a new version of an executable code onto a system without the need to bring the system to a halt,

thereby sacrificing system up time (e.g., [0005]) but does not explicitly disclose a logic module configured to identify incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; the bootstrap module configured to reconcile incompatibilities by changing an initialization order prior to copying the new code image into the memory space occupied by the old code image.

However, in an analogous art of *system and method for implementing a self-activating embedded application*, Rocray discloses:

- a logic module configured to identify incompatibilities (e.g., P. 1, 1st Para - … dynamically resolving compatibility issues between the different components in the system …; 3rd Para - ... comprises a software generic control information file (SGC) … relating to the compatibility of the software components …) between the old code image and the new code image from version information (e.g., P. 7, Lines 11-14 - … identifies which software version ...; P. 8, 2nd Para - … their internal indication of minimum configuration version updated to reflect their incompatibility with the table format identified by the new configuration ...; P. 9, Table 3 – SGC: Component Level Information, the entry of 'version'; P. 10, 1st Para – The version field defines the version of the component), a difference in initialization requirements (e.g., P. 7, Lines 13-14 - … for determining which component are to be used at startup …; P. 11, Lines 6-8 – If such a component were modified through a partial product software update, the whole system would

be re-initialize ...), and a difference in size and location between the old code image and the new code image (e.g., P. 9, Table 3 – SGC: Component Level Information, the entry of 'size');

- the bootstrap module configured to reconcile incompatibilities by changing an initialization order prior to copying the new code image into the memory space occupied by the old code image (e.g., P. 15, Lines 5-8 - Boot code section - ... the boot code of a given processor module ...; P. 19, Lines 6-9 - ... to instruct the system that the alternate bank should be treated as the active bank at the next boot up ...; Lines 12-17 - ... as the result of a software upgrade ... enables the processor modules affected by the update to initialize themselves and boot up)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rocray into the Talati's system to further provide a logic module configured to identify incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; the bootstrap module configured to reconcile incompatibilities by changing an initialization order prior to copying the new code image into the memory space occupied by the old code image in Talati system.

The motivation is that it would further enhance the Talati's system by taking, advancing and/or incorporating the Rocray's system which offers significant advantages that a system and method for dynamically resolving compatibility issues between the different components in the system as once suggested by Rocray (i.e. P. 2, 1st Para)

16. **As to claim 13** (Currently Amended), Talati discloses a system that overlays an old code image with a new code image with minimal interruption of operations being performed by execution of the old code image (e.g., [0011]), the system comprising:

- a memory comprising an old code image (e.g., Fig. 1, element 110; Fig. 2, element 108) and a buffer (e.g., Fig. 1, element 108; Fig. 2, element 110) configured to store a new code image;
- a processor executing instructions of the old code image to perform one or more operations (e.g., Fig. 1, element 106), the processor configured to execute instructions of the old code image and the new code image (e.g., [0011]; [0032], lines 1-5);
- a data structure configured to store an old code image pointer (e.g., Fig. 2, elements 204, 208, and 212; [0023], lines 4-10) and a new code image pointer (e.g., Fig. 2, elements 206, 210, and 214; [0023], lines 11-16);
- wherein, in response to an interrupt, the processor begins executing bootstrap code within the new code image (e.g., [0040])

Further, Talati discloses a method and apparatus for downloading a new version of an executable code onto a system without the need to bring the system to a halt, thereby sacrificing system up time (e.g., [0005]) but does not explicitly disclose identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, the bootstrap code

configured to reconcile incompatibilities between the old code image and the new code image by changing an initialization order.

However, in an analogous art of *system and method for implementing a self-activating embedded application*, Rocray discloses:

- identifying incompatibilities (e.g., P. 1, 1st Para - ... dynamically resolving compatibility issues between the different components in the system ...; 3rd Para - ... comprises a software generic control information file (SGC) ... relating to the compatibility of the software components ...) between the old code image and the new code image from version information (e.g., P. 7, Lines 11-14 - ... identifies which software version ...; P. 8, 2nd Para - ... their internal indication of minimum configuration version updated to reflect their incompatibility with the table format identified by the new configuration ...; P. 9, Table 3 – SGC: Component Level Information, the entry of 'version'; P. 10, 1st Para – The version field defines the version of the component), a difference in initialization requirements (e.g., P. 7, Lines 13-14 - ... for determining which component are to be used at startup ...; P. 11, Lines 6-8 – If such a component were modified through a partial product software update, the whole system would be re-initialize ...), and a difference in size and location between the old code image and the new code image (e.g., P. 9, Table 3 – SGC: Component Level Information, the entry of 'size'),
- the bootstrap code configured to reconcile incompatibilities between the old code image and the new code image by changing an initialization order (e.g.,

P. 15, Lines 5-8 - Boot code section - ... the boot code of a given processor module ...; P. 19, Lines 6-9 - ... to instruct the system that the alternate bank should be treated as the active bank at the next boot up ...; Lines 12-17 - ... as the result of a software upgrade ... enables the processor modules affected by the update to initialize themselves and boot up)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rocray into the Talati's system to further provide identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, the bootstrap code configured to reconcile incompatibilities between the old code image and the new code image by changing an initialization orderr in Talati system.

The motivation is that it would further enhance the Talati's system by taking, advancing and/or incorporating the Rocray's system which offers significant advantages that a system and method for dynamically resolving compatibility issues between the different components in the system as once suggested by Rocray (i.e. P. 2, 1st Para)

17. **As to claim 20** (Currently Amended), Talati discloses a method for updating a code image (e.g., Fig. 2, Copier copies new code), comprising:

- loading a new code image into a temporary memory location (e.g., Fig. 1, element 108; [0009], staging area) separate from a memory space (e.g., Fig. 1,

element 110; [0009], line 2, runtime area) occupied by and used by an old code image (e.g., Fig. 2, elements 218, 110; [0046], lines 1-8; [0047], lines 1-2);

- copying the new code image into the memory space occupied by the old code image (e.g., Fig. 2, element 202; Fig. 3, element 302)

Further, Talati discloses a method and apparatus for downloading a new version of an executable code onto a system without the need to bring the system to a halt, thereby sacrificing system up time (e.g., [0005]) but does not explicitly disclose identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; reconciling incompatibilities by changing an initialization order using bootstrap code of the new code image.

However, in an analogous art of *system and method for implementing a self-activating embedded application*, Rocray discloses:

- identifying incompatibilities between the old code image and the new code image from version information (e.g., P. 7, Lines 11-14 - … identifies which software version ...; P. 8, 2nd Para - … their internal indication of minimum configuration version updated to reflect their incompatibility with the table format identified by the new configuration ...; P. 9, Table 3 – SGC: Component Level Information, the entry of 'version'; P. 10, 1st Para – The version field defines the version of the component), a difference in initialization requirements (e.g., P. 7, Lines 13-14 - … for determining which

component are to be used at startup ...; P. 11, Lines 6-8 – If such a component were modified through a partial product software update, the whole system would be re-initialize ...), and a difference in size and location (e.g., P. 9, Table 3 – SGC: Component Level Information, the entry of 'size') between the old code image and the new code image;

- reconciling incompatibilities by changing an initialization order using bootstrap code of the new code image (e.g., P. 15, Lines 5-8 - Boot code section - ... the boot code of a given processor module ...; P. 19, Lines 6-9 - ... to instruct the system that the alternate bank should be treated as the active bank at the next boot up ...; Lines 12-17 - ... as the result of a software upgrade ... enables the processor modules affected by the update to initialize themselves and boot up)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rocray into the Talati's system to further provide identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; reconciling incompatibilities by changing an initialization order using bootstrap code of the new code image in Talati system.

The motivation is that it would further enhance the Talati's system by taking, advancing and/or incorporating the Rocray's system which offers significant advantages

that a system and method for dynamically resolving compatibility issues between the different components in the system as once suggested by Rocray (i.e. P. 2, 1st Para)

18. **As to claim 21** (original) (incorporating the rejection in claim 20), Talati discloses the method wherein the old code image is updated substantially concurrently with execution of regular computer operations (e.g., [0011]; [0014])

19. **As to claim 22** (original) (incorporating the rejection in claim 20), Talati discloses the method further comprising initiating execution of a run-time segment of the new code image (e.g., [0049])

20. **As to claim 26** (Currently Amended) (incorporating the rejection in claim 20), Talati discloses the method, wherein reconciling the incompatibilities further comprises associating persistent data of with the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image (e.g., [0011]; [0012])

21. **As to claim 29** (Currently Amended), Talati discloses an apparatus for updating a code image (e.g., [0014], lines 2-3; Fig. 2, copier copies new code; [0047], lines 1-2), the apparatus comprising:

- means for loading a new code image (e.g., Fig. 2, element 102) into a temporary memory location (e.g., Fig. 1, element 108; [0009], staging area) separate from a

memory space (e.g., Fig. 1, element 110) occupied by and used by an old code image;

- means for copying the new code image into the memory space occupied by the old code image (e.g., Fig. 2, element 202; Fig. 3, element 302)

Further, Talati discloses a method and apparatus for downloading a new version of an executable code onto a system without the need to bring the system to a halt, thereby sacrificing system up time (e.g., [0005]) but does not explicitly disclose means for identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; means for reconciling the incompatibilities by changing an initialization order using bootstrap code of the new code image.

However, in an analogous art of *system and method for implementing a self-activating embedded application*, Rocray discloses:

- means for identifying incompatibilities (e.g., P. 1, 1st Para - ... dynamically resolving compatibility issues between the different components in the system ...; 3rd Para - ... comprises a software generic control information file (SGC) ... relating to the compatibility of the software components ...) between the old code image and the new code image from version information (e.g., P. 7, Lines 11-14 - ... identifies which software version ...; P. 8, 2nd Para - ... their internal indication of minimum configuration version updated to reflect their incompatibility with the table format identified by the new configuration ...; P.

9, Table 3 – SGC: Component Level Information, the entry of 'version'; P. 10, 1st Para – The version field defines the version of the component), a difference in initialization requirements (e.g., P. 7, Lines 13-14 - ... for determining which component are to be used at startup ...; P. 11, Lines 6-8 – If such a component were modified through a partial product software update, the whole system would be re-initialize ...), and a difference in size and location between the old code image and the new code image (e.g., P. 9, Table 3 – SGC: Component Level Information, the entry of 'size');

- means for reconciling the incompatibilities by changing an initialization order using bootstrap code of the new code image (e.g., P. 15, Lines 5-8 - Boot code section - ... the boot code of a given processor module ...; P. 19, Lines 6-9 - ... to instruct the system that the alternate bank should be treated as the active bank at the next boot up ...; Lines 12-17 - ... as the result of a software upgrade ... enables the processor modules affected by the update to initialize themselves and boot up)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rocray into the Talati's system to further provide means for identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; means for reconciling the incompatibilities by changing an initialization order using bootstrap code of the new code image in Talati system.

The motivation is that it would further enhance the Talati's system by taking, advancing and/or incorporating the Rocray's system which offers significant advantages that a system and method for dynamically resolving compatibility issues between the different components in the system as once suggested by Rocray (i.e. P. 2, 1st Para)

22. **As to claim 30** (Currently Amended), Talati discloses an article of manufacture comprising a program storage medium readable by a processor and embodying one or more instructions executable by a processor to perform a method for updating a code image (e.g., Fig. 1), the method comprising:

- loading a new code image into a temporary memory location separate from a memory space occupied by and used by an old code image (e.g., Fig. 2, elements 218, 110; [0046], lines 1-8; [0047], lines 1-2);
- copying the new code image into the memory space occupied by the old code image (e.g., Fig. 2, element 202; Fig. 3, element 302)

Further, Talati discloses a method and apparatus for downloading a new version of an executable code onto a system without the need to bring the system to a halt, thereby sacrificing system up time (e.g., [0005]) but does not explicitly disclose identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; reconciling the incompatibilities by changing an initialization order using bootstrap code of the new code image.

However, in an analogous art of *system and method for implementing a self-activating embedded application*, Rocray discloses:

- identifying incompatibilities (e.g., P. 1, 1st Para - ... dynamically resolving compatibility issues between the different components in the system ...; 3rd Para - ... comprises a software generic control information file (SGC) ... relating to the compatibility of the software components ...) between the old code image and the new code image from version information (e.g., P. 7, Lines 11-14 - ... identifies which software version ...; P. 8, 2nd Para - ... their internal indication of minimum configuration version updated to reflect their incompatibility with the table format identified by the new configuration ...; P. 9, Table 3 – SGC: Component Level Information, the entry of 'version'; P. 10, 1st Para – The version field defines the version of the component), a difference in initialization requirements (e.g., P. 7, Lines 13-14 - ... for determining which component are to be used at startup ...; P. 11, Lines 6-8 – If such a component were modified through a partial product software update, the whole system would be re-initialize ...), and a difference in size and location between the old code image and the new code image (e.g., P. 9, Table 3 – SGC: Component Level Information, the entry of 'size');
- reconciling the incompatibilities by changing an initialization order using bootstrap code of the new code image (e.g., P. 15, Lines 5-8 - Boot code section - ... the boot code of a given processor module ...; P. 19, Lines 6-9 - ... to instruct the system that the alternate bank should be treated as the active

bank at the next boot up ...; Lines 12-17 - ... as the result of a software upgrade ... enables the processor modules affected by the update to initialize themselves and boot up)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rocray into the Talati's system to further provide identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; reconciling the incompatibilities by changing an initialization order using bootstrap code of the new code image in Talati system.

The motivation is that it would further enhance the Talati's system by taking, advancing and/or incorporating the Rocray's system which offers significant advantages that a system and method for dynamically resolving compatibility issues between the different components in the system as once suggested by Rocray (i.e. P. 2, 1st Para)

23. Claim 5-6, 9, 11-12, 23-25, and 28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Talati in view of Rocray and further in view of Judith E. Schwabe (Pat. No. US 6,986,132 B1) (hereinafter 'Schwabe')

24. **As to claim 5** (Currently Amended) (incorporating the rejection in claim 1), Talati and Rocray do not disclose the apparatus further comprising a logic module configured to access the version information for the old code image and version information for the

new code image and identify an incompatibility based at least in part on a difference between the version information.

However, in an analogous art of *Remote Incremental Program Binary Compatibility Verification Using API Definitions*, Schwabe discloses the apparatus further comprising a logic module configured to access the version information for the old code image and version information for the new code image and identify an incompatibility based at least in part on a difference between the version information (e.g., Fig. 17, element 1440 – version; Fig. 18, element 1470 – version; Fig. 19, element 1515 – verify version of API definition file used during verification is compatible with version of referenced binary file; Fig. 20A – 3. verify backward compatible version with content; Fig. 20C – verify versions using API definitions files, elements of 1600, 1605, and 1610).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Schwabe into the Talati-Rocray's system to further provide the apparatus further comprising a logic module configured to access the version information for the old code image and version information for the new code image and identify an incompatibility based at least in part on a difference between the version information in Talati-Rocray system.

The motivation is that use of the verifier enables verification of a program's integrity and allows the use of an interpreter that does not execute the usual stack monitoring instructions during program execution, thereby greatly accelerating the

program interpretation process as once suggested by Schwabe (i.e. Col. 13, Line 66 through Col. 14, Line 8)

25. **As to claim 6** (Previously Presented) (incorporating the rejection in claim 5), Schwabe discloses the apparatus wherein the bootstrap module is configured to update modules that interface with the new code image based at least in part on a difference between the version information (e.g., Fig. 17, element 1440 – version; Fig. 18, element 1470 – version; Fig. 19, element 1515 – verify version of API definition file used during verification is compatible with version of referenced binary file; Fig. 20A – 3. verify backward compatible version with content; Fig. 20C – verify versions using API definitions files, elements of 1600, 1605, and 1610)

26. **As to claim 9** (original) (incorporating the rejection in claim 1), Schwabe discloses the apparatus wherein at least one of the incompatibilities comprises a difference between data structures used by the old code image and data structures used by the new code image (e.g., Col. 22, Lines 30-33)

27. **As to claim 11** (Currently Amended) (incorporating the rejection in claim 10), Schwabe discloses the apparatus wherein the bootstrap module comprises a conversion module configured to reconcile the incompatibilities base on the version information for the old code image and the new code image and a copy module configured to copy the new code image over the old code image in response to

reconciliation of the incompatibilities (e.g., Fig. 17, element 1440 – version; Fig. 18, element 1470 – version; Fig. 19, element 1515 – verify version of API definition file used during verification is compatible with version of referenced binary file; Fig. 20A – 3. verify backward compatible version with content; Fig. 20C – verify versions using API definitions files, elements of 1600, 1605, and 1610)

28. **As to claim 12** (original) (incorporating the rejection in claim 10), Schwabe discloses the apparatus wherein at least one of the incompatibilities comprises a difference between data structures used by the old code image and data structures used by the new code image (e.g., Col. 22, Lines 30-33)

29. **As to claim 23** (Previously Presented) (incorporating the rejection in claim 20), Schwabe discloses the method wherein the new code image is not copied into the memory space until incompatibilities are reconciled (e.g., Col. 5, Lines 49-53; Col. 10, Lines 40-44; Col. 11, Line 58 through Col. 12, Lines 6; Figs. 15A-15B; Col. 20, Line 64 through Col. 21, Line 4, 14-20; Fig. 17; Col. 22, Lines 4-16; Figs. 20A-20D; Col. 24, Lines 24-32)

30. **As to claim 24** (Currently Amended) (incorporating the rejection in claim 20), Schwabe discloses the method, wherein identifying incompatibilities between the old code image and the new code image further comprises accessing capability information for the old code image and capability information for the new code image and identifying

an incompatibility based at least in part on a difference between the capability information (e.g., Col. 9, Lines 6-11)

31. **As to claim 25** (Previously Presented) (incorporating the rejection in claim 24), Schwabe discloses the method, wherein reconciling the incompatibilities comprising updating modules that interface with the new code image based at least in part on a difference between the capability information (e.g., Col. 9, Lines 6-11)

32. **As to claim 28** (original) (incorporating the rejection in claim 20), Schwabe discloses the method wherein at least one of the incompatibilities comprises a difference between data structures used by the old code image and data structures used by the new code image (e.g., Col. 22, Lines 30-33)

33. Claim 14-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Talati in view of Rocray and further in view of Hiller (Pat. No. US 6,658,659 B2) (hereinafter 'Hiller')

34. **As to claim 14** (original) (incorporating the rejection in claim 13), Talati and Rocray do not disclose the system wherein the bootstrap code overlays the new code image in memory with the old code image in response to reconciliation of the incompatibilities.

However, in an analogous art of *Compatible Version Module Loading*, Hiller discloses the system wherein the bootstrap code overlays the new code image in memory with the old code image in response to reconciliation of the incompatibilities (e.g., Fig. 2A, element 206; Fig. 2B; Col. 3, Lines 42-46; Col. 4, Lines 64-67))

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hiller into the Talati-Rocray's system to further provide the system wherein the bootstrap code overlays the new code image in memory with the old code image in response to reconciliation of the incompatibilities in Talati- Rocray system.

The motivation is that the system wherein the version aware bootstrap code will check and ensure that loaded software modules are compatible with one another and will therefore execute properly as once suggested by Hill (i.e. Abstract, Lines 10-13)

35. **As to claim 15** (original) (incorporating the rejection in claim 14), Talati discloses the system wherein in response to the interrupt, the processor executes an update module of the old code image that loads the new code image into the buffer (e.g., Fig. 3, step 302; [0040]; [0041])

36. **As to claim 16** (original) (incorporating the rejection in claim 15), Talati discloses the system wherein the update module stores the old code image pointer (e.g., Fig. 2, elements 204, 208, and 212; [0023], lines 4-10) and the new code image pointer (e.g., Fig. 2, elements 206, 210, and 214; [0023], lines 11-16) in the data structure.

37. **As to claim 17** (original) (incorporating the rejection in claim 16), Talati discloses the system of wherein the update module reads a new code image header identified by the new code image pointer (e.g., Fig. 2, elements 206, 210; [0034]) to determine the location of the bootstrap code within the new code image (e.g., [0037], lines 5-7)

38. Claim 18-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Talati in view of Rocray, Hiller, and Schwabe.

39. **As to claim 18** (original) (incorporating the rejection in claim 17), Talati, Rocray and Hiller do not disclose the system, wherein the bootstrap code further reconciles the incompatibilities by updating modules that interface with the new code image.

However, in an analogous art of *Remote Incremental Program Binary Compatibility Verification Using API Definitions*, Schwabe discloses the system, wherein the bootstrap code further reconciles the incompatibilities by updating modules that interface with the new code image (e.g., Col. 9, Lines 6-11)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Schwabe into the Talati-Rocray-Hiller's system to further provide the system, wherein the bootstrap code further reconciles the incompatibilities by updating modules that interface with the new code image in Talati-Rocray-Hiller system.

The motivation is that use of the verifier enables verification of a program's integrity and allows the use of an interpreter that does not execute the usual stack monitoring instructions during program execution, thereby greatly accelerating the program interpretation process as once suggested by Schwabe (i.e. Col. 13, Line 66 through Col. 14, Line 8)

40. **As to claim 19** (Currently Amended) (incorporating the rejection in claim 18), Schwabe discloses the system wherein the bootstrap code further reconciles the incompatibilities by associating persistent data of the old code image with the new code image (e.g., Fig. 17, element 1440 – version; Fig. 18, element 1470 – version; Fig. 19, element 1515 – verify version of API definition file used during verification is compatible with version of referenced binary file; Fig. 20A – 3. verify backward compatible version with content; Fig. 20C – verify versions using API definitions files, elements of 1600, 1605, and 1610)

Conclusion

41. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Examiner, Art Unit 2192
September 26, 2008

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192